MLOps Workflow Documentation

Objective

The objective of this project is to come up with a MLOPS tool that includes the whole process of a machine learning model from data ingestion, training to deployment. In this case the user could easily reuse the trained model and better handle the CI/CD process. The data used here is from a bank, where the task is to predict if a customer has a deposit or is likely to deposit money.

Structure

Our system architecture includes workflows related to modelling, interfaces for the users, test and validation of models as well as deployment through a Docker container.

First, for modelling, our architecture includes a folder for the dataset (in this case the bank dataset). There is also a folder incorporating the core machine learning functions with a reusable pipeline that includes data preprocessing. The models used are random forests, logistic regression and support vector machines. This /ml folder includes ml/functions.py for data ingestion, preprocessing and evaluation and ml/models.py for model selection and model training. The trained models are then stored in pickle files within the checkpoint folder.

Several interfaces are available. For model training and evaluation, the users could use a command line interface that supports a command to list all available models and one to train and evaluate a model. Similarly, the /FastApi folder contains an API to enable the same tasks through HTTP requests.

For testing, there is a folder with Python’s unittest framework. The tests are done on ml/functions.py to ensure data preprocessing and model evaluations are done correctly and on ml/models.py to validate model selection and training. For instance, we can test whether data splitting between training and testing sets was successful.

Finally, there is a user interface through a Streamlit application to interact with the trained models. This should be particularly helpful for non-technical audiences who need to see the model’s predictions. A Docker container can be used to launch the Streamlit application.

User Guide

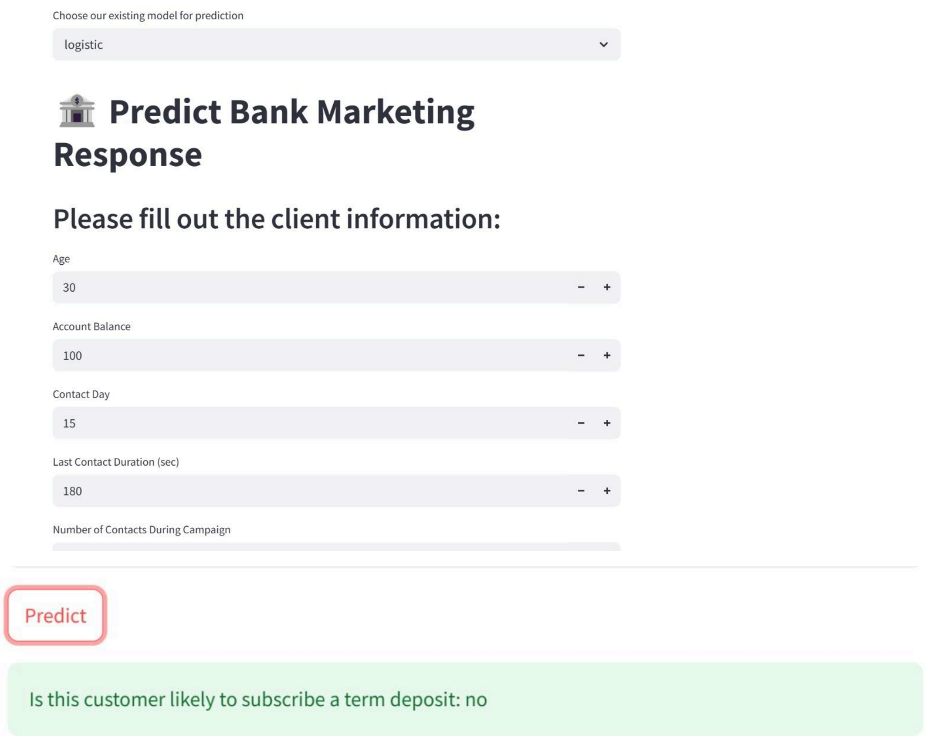
To ensure usability of our tool, dependencies are managed and explanations are provided. The users should have their current working directory inside the MLOps_group folder. This folder includes a README that explains the tool and its components. Dependencies are managed through a requirements.txt file and a pyproject.toml file. The requirements file mainly concerns the deployment of the Streamlit application and the containerization with Docker. These can then be easily installed. The pyprojects file ensures that dependencies are managed in a flexible manner as they can be installed thanks to hatchling.

Future Work

This tool could be further improved by being more flexible when it comes to the dataset (for instance, dealing with live data). Other improvements could include enhanced monitoring, automated retraining pipelines and cloud deployment.

Appendix

1, The streamlit interface: allow user to manipulate the model and compare the different performances of different models with dynamic and user-friendly interfaces



Choose our existing model for prediction

logistic

🏦 Predict Bank Marketing Response

Please fill out the client information:

Age

30

Account Balance

100

Contact Day

15

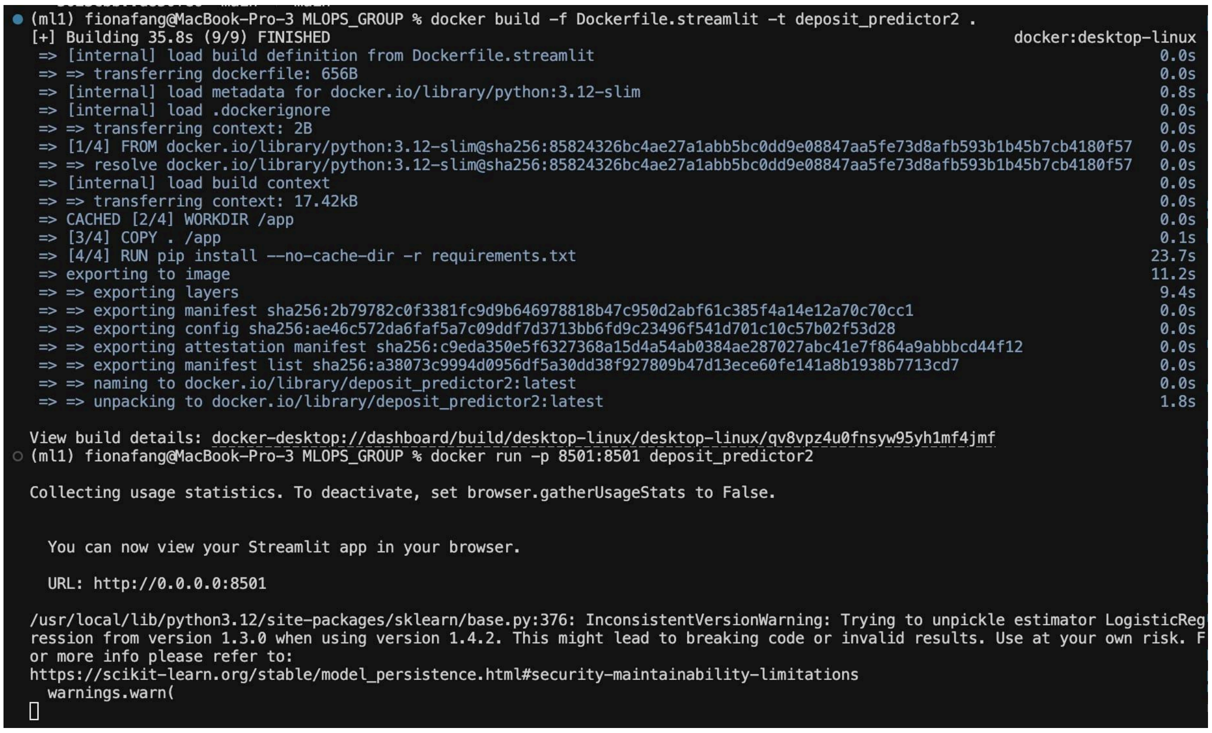
Last Contact Duration (sec)

180

Number of Contacts During Campaign

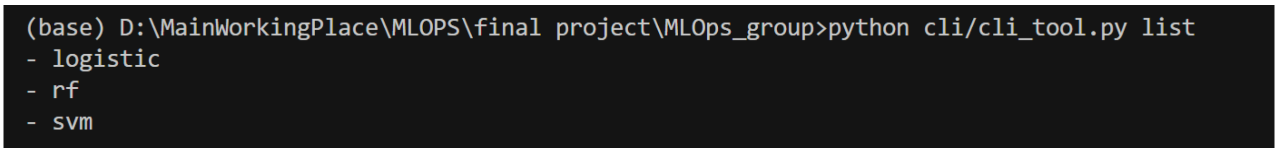
Is this customer likely to subscribe a term deposit: no

And below is a screenshot of the successful implementation of its containerization using docker.

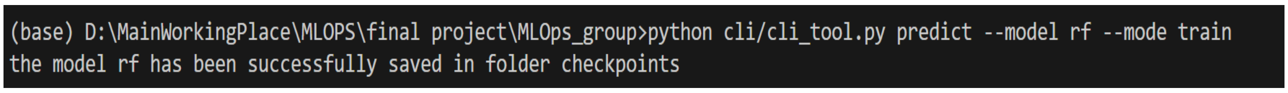


```
● (ml1) fionafang@MacBook-Pro-3 MLOPS_GROUP % docker build -f Dockerfile.streamlit -t deposit_predictor2 .
  [+] Building 35.8s (9/9) FINISHED                                                     docker:desktop-linux
   => [internal] load build definition from Dockerfile.streamlit                                       0.0s
   => => transferring dockerfile: 656B                                                                 0.0s
   => [internal] load metadata for docker.io/library/python:3.12-slim                                  0.8s
   => [internal] load .dockerignore                                                                    0.0s
   => => transferring context: 2B                                                                      0.0s
   => [1/4] FROM docker.io/library/python:3.12-slim@sha256:85824326bc4ae27a1abb5bc0dd9e08847aa5fe73d8afb593b1b45b7cb4180f57  0.0s
   => => resolve docker.io/library/python:3.12-slim@sha256:85824326bc4ae27a1abb5bc0dd9e08847aa5fe73d8afb593b1b45b7cb4180f57  0.0s
   => [internal] load build context                                                                    0.0s
   => => transferring context: 17.42kB                                                                 0.0s
   => CACHED [2/4] WORKDIR /app                                                                        0.0s
   => [3/4] COPY . /app                                                                                0.1s
   => [4/4] RUN pip install --no-cache-dir -r requirements.txt                                        23.7s
   => exporting to image                                                                              11.2s
   => => exporting layers                                                                              9.4s
   => => exporting manifest sha256:2b79782c0f3381fc9d9b646978818b47c950d2abf61c385f4a14e12a70c70cc1    0.0s
   => => exporting config sha256:ae46c572da6faf5a7c09ddf7d3713bb6fd9c23496f541d701c10c57b02f53d28      0.0s
   => => exporting attestation manifest sha256:c9eda350e5f6327368a15d4a54ab0384ae287027abc41e7f864a9abbbcd44f12  0.0s
   => => exporting manifest list sha256:a38073c9994d0956df5a30dd38f927809b47d13ece60fe141a8b1938b7713cd7  0.0s
   => => naming to docker.io/library/deposit_predictor2:latest                                         0.0s
   => => unpacking to docker.io/library/deposit_predictor2:latest                                      1.8s

  View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/qv8vpz4u0fnsyw95yh1mf4jmf
○ (ml1) fionafang@MacBook-Pro-3 MLOPS_GROUP % docker run -p 8501:8501 deposit_predictor2

  Collecting usage statistics. To deactivate, set browser.gatherUsageStats to False.


    You can now view your Streamlit app in your browser.

    URL: http://0.0.0.0:8501

  /usr/local/lib/python3.12/site-packages/sklearn/base.py:376: InconsistentVersionWarning: Trying to unpickle estimator LogisticReg
  ression from version 1.3.0 when using version 1.4.2. This might lead to breaking code or invalid results. Use at your own risk. F
  or more info please refer to:
  https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
    warnings.warn(
  ▯
```

2, CLI Interface: by executing the file cli/cli_tool.py, the user could pass different commands as well as arguments to run the model/ check available models

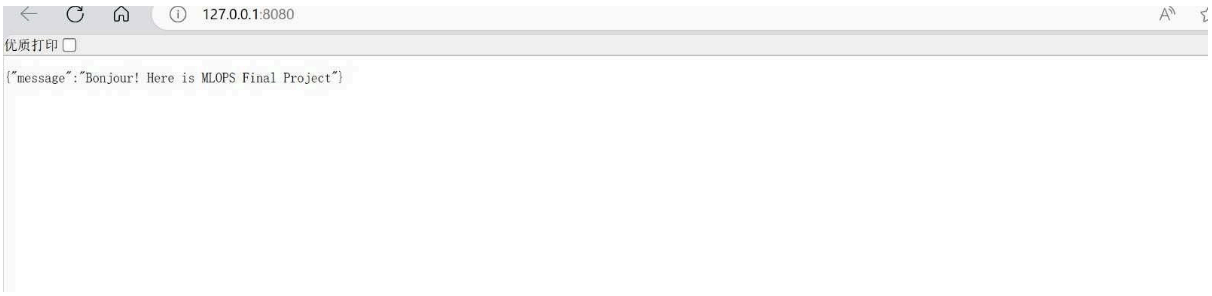


```
(base) D:\MainWorkingPlace\MLOPS\final project\MLOps_group>python cli/cli_tool.py list
- logistic
- rf
- svm
```



```
(base) D:\MainWorkingPlace\MLOPS\final project\MLOps_group>python cli/cli_tool.py predict --model rf --mode train
the model rf has been successfully saved in folder checkpoints
```

3, FastAPI: here is a screenshot of successful implementation of fastapi:
root (host address: 127.0.0.1, the port is 8080)

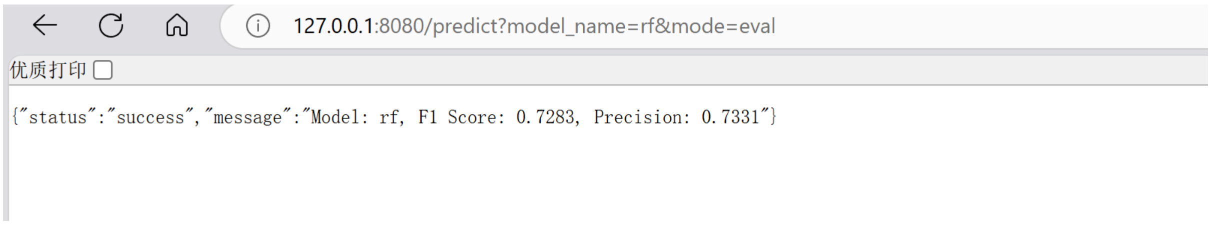


127.0.0.1:8080

优质打印

```
{"message":"Bonjour! Here is MLOPS Final Project"}
```

endpoint: predict. The user can also access other services (run model) by appending url parameters like model_name and mode.



127.0.0.1:8080/predict?model_name=rf&mode=eval

优质打印

```
{"status":"success","message":"Model: rf, F1 Score: 0.7283, Precision: 0.7331"}
```